Another Computer
Product from

STACK

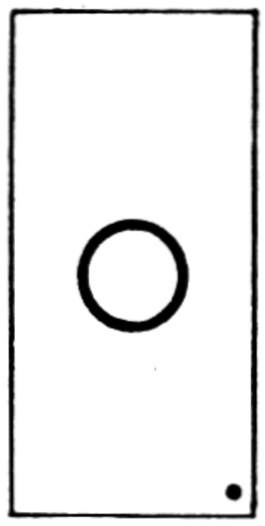
VICkit

Stack Computer Services Ltd
290-298 Derby Road, Bootle
Liverpool L20 8LN.
Tel 051 933 5511
Telex 627026

Installing the VICKIT

First touch any safely earthed piece of metal to discharge any static electrical charge in your body. Carefully unpack the chip and examine the pins. If any pins are bent or out of alignment straighten them using narrow pointed pliers GENTLY. I.C. (chip) pins are soft and fragile and fracture very easily with bending. All of the chips are tested using zero insertion force sockets before dispatch and are packed in rigid pin-protecting tubes.

The Stack VICKIT EPROM



This dot marks pin 1

Note that the chip MUST be correctly orientated in its socket, which should be set to be located at $B000 in order to be addressed correctly by the VIC*20 (see the instructions for the expansion unit you are using to hold the VICKIT in order to discover how to achieve this). REVERSAL of the orientation of the chip will probably DESTROY the chip instantly by breaking down the stored program throughout the chip or damaging its internal circuitry.

Great care must be used in inserting the chip into its socket. All of the pins must be aligned simultaneously and slow firm pressure used to seat the chip. If you bend the pins at this stage they will probably break off when you try to straighten them so DON'T BEND THEM.

If you haven't done this before and don't feel confident then get assistance from someone who has experience of chip insertion.

---

*VIC is a registered trade mark of CBM Ltd.

Disclaimer

Stack Computer Services Ltd have taken every precaution to ensure that the information herein is accurate but reserve the right to alter or amend future manuals without notice. No liability expressed or implied is accepted in respect of any loss or damage in respect of the use of this device.

Copyright

Stack Computer Services Ltd,
290-298 Derby Road,
Bootle,
Liverpool 20.
051 933 5511
Telex 627026 (StackG)

Bugs

At the time of writing this manual it is believed that all known problems with the VICKIT have been documented but the author will be grateful to receive correspondence from any persons who believe that they have found unacceptable problems.

The VICKIT offers facilities to the BASIC programmer on the VIC-20 similar to those provided by the BASIC Programmer's Toolkit tm* available for the PET. A number of useful routines have been written specifically for the VIC-20 and placed in a 2K (2048) byte EPROM, which does not occupy any of the VIC-20's programmable memory.

Programmers with experience of the BASIC Programmer's Toolkit for the PET should read the sections on installing, enabling and disabling the VICKIT and then turn to the section at the end of this manual describing the differences between the VICKIT and the Toolkit.

Programmers unfamiliar with the Toolkit should read the rest of this manual thoroughly in order to get the best from their VICKIT.

When the VICKIT has been installed and enabled your VIC-20 will work as before but will have 9 new commands added to the VIC-20 BASIC language. These commands (they are not statements and cannot be included in a BASIC program) are:

AUTO	which produces line number for you AUTOmatically while entering a program into the VIC-20
DELETE	which lets you DELETE a range of line numbers from your program with one statement, rather than by deleting one line at a time
DUMP	which DUMPs the value of all simple (i.e. not array) variables used in your program to the screen, where you can alter them if you wish
FIND	which lets you FIND all the occurences of a given string or BASIC statement in your program
HELP	which, if you get an error while running your program, HELPs you to correct it by displaying the line in error and indicating where on the line the error occurred
OFF	which is used to switch OFF either of the two tracing commands (TRACE and STEP) when their use is no longer required
RENUMBER	which allows you to RENUMBER the line numbers in your program, either to gain space for more lines or simply to make the program look neater
STEP	which enables a program to be run one statement at a time, displaying the line number currently being executed and allowing you to STEP through your program statement by statement using the shift key on the VIC-20
TRACE	which functions like STEP except that you can let your program run quickly to a section of interest and then slow execution down, TRACEing the last 5 lines executed on the screen.

A final addition to the VIC-20 BASIC is that LIST, when entered as a command, works at your speed rather than at the VIC-20's. As with the normal LIST command you can LIST a section of your program or even the whole program but after each line is displayed on the screen you can either stop the LISTing altogether (using the STOP key) or move on to the next line by pressing the SPACE key. Hold the SPACE key down and the LISTing proceeds at normal, let the SPACE key go and the LISTing halts temporarily should you see a line of interest. Make notes, discover what is wrong, find out how something works and by pressing the SPACE key again move on. You no longer need the reflexes of a racing driver to keep interesting lines on the screen.

* BASIC Programmer's Toolkit is a trademark of Palo Alto ICs, a division of Nestar Systems, Inc.

Enabling the VICKIT

When your VICKIT has been installed turn on the VIC-20. If the CBM BASIC message does not appear TURN OFF AT ONCE and recheck the steps you took during the installation. Assuming that the VIC-20 has started as normal enter the following command:

```
        SYS(11*4096)
or      SYS(45056)
```

and press RETURN. You should then see a copyright notice appear on the screen followed by the READY prompt from the VIC-20 and the flashing cursor. If this does not happen TURN OFF AT ONCE and again check your installation of the VICKIT. If you have a board which enables one or more sets or ROMs or EPROMs to be located at the same addresses in the VIC-20 check that the socket containing the VICKIT is one of the ones currently enabled and selected and that the VICKIT is located in a socket addressed at $B000.

If the VICKIT still does not work you should contact the supplier of the VICKIT for assistance.

Disabling the VICKIT

Having just enabled the VICKIT it may seem strange to discuss how to disable the VICKIT but such a step may be necessary, for example, should you want to switch to a different set of ROMs or EPROMs with a ROM SWITCH BOARD. You might also want to add some other ROMs or EPROMs to the VIC-20 which conflict with the
* VICKIT. If you switch over from the VICKIT while it is still disabled you will most likely lose control of the VIC-20 until you use RUN/STOP and RESTORE to reset the VIC-20. In some cases you might lose control altogether and have to switch the VIC-20 off and on again.

Assuming that the VICKIT has been enabled and you have the READY prompt and flashing cursor from the VIC-20 enter the following command:

```
        SYS(58459)
```

and press RETURN. The READY prompt will reappear and you will find that none of the VICKIT commands will work.

Note that both the enabling and disabling SYS commands may be given inside a program but the VICKIT commands may not be so included.

A note on the manual

Throughout the manual a sunburst sign (*) is used to indicate a section of particular importance, normally a section where misuse of the VICKIT can result in damage to your program.

Using the VICKIT

The VICKIT is normally used in three stages of program development, firstly the actual entering of the program, secondly the testing of the program and thirdly the tidying up of the program to its final version.

This manual illustrates these three stages by leading you through the development of a typical program using the VICKIT. When a new VICKIT command is used it will be explained in detail there and then with examples of its use on the developing program.

The program illustrated here is one to find the day of the week that a certain date fell on and will work for all dates after 1752, when there was a change in the calendar. The actual program comes from the Independent Pet Users' Group magazine, volume 3 number 6.

Firstly we will enter the program, starting at line 10, with the second line being 20, the third 30 and so on. To save us from having to enter the line number of each line (possibly getting it wrong and deleting a needed line) we use the AUTO command.

AUTO There are two forms of the AUTO command, the first when you specify the line number that you want to start with only and the second when you specify both the starting line number and the step between line numbers.

To use the first form enter the command:

AUTO <starting line number>

and press RETURN. For example to start generating line number for our program simply enter:

AUTO 10 or just AUTO

and press RETURN. The VICKIT will respond with:

```
10■
```

where ■ represents the flashing cursor. You can now enter the first line of the program by entering:

```
DAY$(1)="SUNDAY":DAY$(2)="MONDAY"
```

and pressing RETURN. The VIC-20 will take in line 10 to your program and the VICKIT will print the next line number as follows (since you did not use the second form of AUTO the VICKIT used 10 as the step between line numbers):

```
20■
```

You can now enter the second line of the program by typing:

```
DAY$(3)="TUESDAY":DAY$(4)="WEDNESDAY":DAY$(5)="THURSDAY"
```

and pressing RETURN. Line 20 will be taken in by the VIC-20 and the VICKIT will print the next line number:

```
30■
```

You could of course now go on and insert the rest of the program but first let us look at AUTO in more detail.

First of all, to stop AUTO from operating simply press RETURN in response to a line number. Carrying on with our example simply pressing RETURN for line 30 will not result in the VICKIT responding with 40 but will switch the AUTO routine off. Since typing a line number followed immediately by RETURN is the standard VIC-20 way to delete a line we need some other way of turning AUTO off if there is the least chance that we will delete an important line. This second way is accomplished by using the DEL key to delete the line number before pressing RETURN.

The second form of the AUTO command can be demonstrated by typing in some lines of rubbish starting at line 30 and on lines 35, 40 and 45. The AUTO command to do this has the following form:

AUTO <starting line number>,<line number step>

and so you should enter the following command:

AUTO 30,5

and press RETURN. The VICKIT will respond with:

30 ■

to which you should enter anything that you wish (we will be removing these lines in the next section). Pressing RETURN at the end of entering line 30 will produce the next line number:

35 ■

to which you should add your second line of 'anything', press RETURN and so on.

When you have added line 45, pressed RETURN and got VICKIT to prompt with line number 50 simply press RETURN (or using the DEL key delete the 50 and press RETURN). We now have line 30, 35, 40 and 45 which we do not want in our final program. We could delete them line by line in the standard VIC-20 manner but using the VICKIT's DELETE command makes things much easier.

DELETE

The DELETE command has a form, or rather number of forms, rather like the VIC-20 command LIST for listing a section of program. If we wanted to LIST the unwanted lines we would enter:

LIST 30-45

or

LIST 30-

since the unwanted lines go from line 30 to the end of the program. To DELETE these lines we give the command:

DELETE 30-45

or

DELETE 30-

and press RETURN. The unwanted lines would then have disappeared. Note that we can only use the second form since the lines that we want to DELETE run from line 30 to the end of the program. Had there been lines after line 45 that we wanted to preserve the first form of DELETE would have to be used.

Like LIST we could also DELETE line number from the beginning of the program to a certain line number, e.g.:

DELETE -20

followed by RETURN would DELETE what is left of the sample program (if you actually gave this command go back and enter the DELETEd lines again otherwise your program will not work).

Thus the form of the DELETE command is exactly like that of LIST

* (although using DELETE 150 etc will DELETE from line 150 to the end of the program. Simply typing 150 and RETURN is easier). with one exception. Entering DELETE on its own will not DELETE the entire program since there is a perfectly good command (NEW) to do this already in the VIC-20 BASIC.

* DELETE n-m where n is greater than m will corrupt your program. Always keep an up-to-date backup copy of your program before using DELETE.

Back to entering the program again. Restart AUTO at line 30 with a line number step of 10 using the command:

AUTO 30,10

and pressing RETURN. Enter the following lines exactly as written,although not of

course the line numbers:

```
30 DAY$(6)="FRIDAY":DAY$(7)="SATURDAY":D=0:M=0:Y=0
40 INPUT "DATE EG 20,11,1981";DAY,MNTH,YEAR:IF YEAR>1753 THEN 60
50 PRINT "DATE MUST NOT BE PRIORTO 1753":GOTO 40
60 K=INT(0.6+(1/MNTH)):L=YEAR-K:P=K/100
70 Z=INT(13*(MNTH+12*K+1)/5)+INT((5*L)/4)-INT(P)+INT(P/4)+DAY-1
80 Z=Z-(7*INT(Z/7))+1
90 PRINT "THE DATE";D;M;Y:PRINT "WAS OR WILL BE":PRINT "A ";DAY$(Z
100 PRINT "ANOTHER DATE (YES OR":PRINT "NO)";:INPUT ANS$
110 IF ANS$="YES" THEN PRINT:GOTO 40
120 IF ANS$="NO" THEN 150:REM EXIT
130 PRINT "DO IT PROPERLY":GOTO 100
140 PRINT "GOODBYE":END
150
```

Press RETURN only for line 150 to stop AUTO generating line numbers.

Now RUN the program and see what happens. Enter a few dates whose day is known to you and you should find a number of errors. The most obvious one is that the program do not work since it does not tell you the correct day in all cases. The second error is that the date is printed on the screen as 0, 0 and 0, rather than as the date that you typed in. Also if you accidently added some errors of your own you might find the program not even printing an answer.

Now in programming there are two main sorts of errors. The first is an error in the logic behind the statements in the program which means that although the program RUNs the answers that it produces are not the correct ones. The second type is an error in the actual program statements which the BASIC interpreter can detect. In this case a helpful error message is produced by the VIC-20 giving not only the type of error but the line on which it occurred. In general the first type of error is the harder to cure but the VICKIT provides three commands to help you while in the second type there is only HELP for you. The sample program has both types of errors and we will use all the relevant facilities of the VICKIT to track them down.

To find the first kind of error the commands DUMP, STEP and TRACE are used and we shall look at the program first using the DUMP command. Wait until you are asked by the program for the response YES or NO in line 100:

```
ANOTHER DATE (YES OR
NO)?
```

before pressing the SHIFT and RUN/STOP keys to break out of the program and get the READY prompt back again.

DUMP To display the values of all variables used by the program (all simple variables that is, array variables are no DUMPed by the VICKIT) simply give the command:

```
DUMP
```

and press RETURN. You will find that the variables used in the program so far will be listed, one to a line, on the screen. After each one is listed you can either press the STOP key to stop the DUMP or press the SPACE bar to continue with the listing. With the simple program above you will find the following being printed:

```
D  =0
M  =0
Y  =0
DA =
MN =
YE =
K  =
L  =
P  =
Z  =
AN$="LOAD"
```

Variables except D, M and Y will have values that will depend on what answers you gave to the questions. Note that only the first two characters of variable names are given in the DUMP, e.g. DA not DAY, and that string variables are shown with their value inside inverted commas so that cursor and colour control characters will show up properly and can be altered using the screen editing facility. All types (real, integer and string) of variables are DUMPed by the VICKIT and you can alter any values on the screen as if they were program lines. Thus a program can be stopped halfway through its RUN, its variables examined and possibly altered, then the program CONTinued.

Looking at the list of DUMPed variables you can see that DA, MN and YE have the values that you last gave them in response to the INPUT statement of line 40. Why then are they printed out as 0, 0 and 0? Looking down the list you can see that D, M and Y all have the value 0, and if we list line 90 (don't forget to press the space bar after each line is printed) we can see that we printed out D, M and Y rather than DA, MN and YE.

Alter line 90 to print DAY;MNTH;YEAR and RUN the program again. Now you should find the date being printed correctly to the screen but the day of the week being printed out will probably still be wrong.

The last error in the actual calculation part of the program is of the type which you are unlikely to find using the VICKIT. The statement P=K/100 should in fact be P=L/100. Since the statement is correct BASIC it will not cause a BASIC error so the only way that we could discover the error is if we knew what the value of P was likely to be. In fact it is usually the number of hundred year periods from 0 to the date input, e.g. for 1952 P would equal 19. DUMPing the variables would help us discover that P was set to a value less than 1 so would give the clue that the problem was with the statement which set P to a value, i.e. a statement beginning with P=. To find the statement in the program which sets P to a value we can use the fourth of the VICKIT's commands: FIND.

FIND Assume that we do not know on which line the statement P=K/100 occurs. To FIND which line it is on give the VICKIT command:

FIND P=

* and press RETURN. Note that there must be a space after FIND.

The VICKIT will then list the lines in the program which contain the section of programming P=. After each line is listed you must remember to press the SPACE bar in order to carry on to list the next line. The FIND command is of course much more useful in a larger program. For example, suppose that the program you are working on requires a variable to be defined to hold a value temporarily. Can you use the variable DT? You could look through the entire program for any uses of the variable DT, or if you have been efficient you would have written down each variable name as you used it. With the VICKIT you have a third option, which enables you to devote your energies to programming, rather than housekeeping. Simply give the VICKIT command:

FIND DT

and press RETURN. Any lines of BASIC programming containing DT will be listed on the screen. If no lines are listed then you can use the variable DT freely.

As with LIST you can press STOP if the information that you require has appeared on the screen.

You can also restrict the area of your program that VICKIT will search when using FIND. After giving the section of programming that you are interested in type a comma followed by a range of line

numbers in the same form as you use for LIST and DELETE. Thus to FIND any sections of programming which use P= in the lines 10-150 simply type the following VICKIT command:

```
FIND P=,10-150
```

and press RETURN. As with DELETE you can omit either the first or the second number, e.g.

```
FIND P=,10-
```

or

```
FIND P=,-150
```

So far we have just looked for BASIC statments, or rather parts of them. There is another form of FIND which will look for strings in your BASIC program, which will occur in a number of places. Most obviously they will appear inside quotes, e.g. DATE E.G. in line 40. However they will also occur in REM and DATA statements for the following reason: when you type in a line of BASIC into the VIC-20 it will normally convert BASIC keywords and characters, IF, PRINT, THEN, +, = etc, into single characters called tokens. These tokens are not usually typed directly from the keyboard and since the word PRINT is converted into a token for example, the five locations that would be used storing PRINT as five characters are reduced to one, for storing the token for PRINT. Tokens also speed the execution of a program up. However, when you type PRINT within quotes, or in a DATA or REM statement, this crunching into tokens is not performed, and PRINT would be stored in five characters.

Because of tokens you can see that there could be two forms of the characters P=. The first would be when P= has appeared within quotes, or in a REM or DATA statment, when = won't have been replaced by its token. The second will be when P= has appeared as a BASIC statement and = has been replaced by its token. So far we have been FINDing the tokenised form but the VICKIT will also FIND the un-tokenised form providing that you enclose the string to be searched for within quotes, e.g.:

```
FIND "P="
```

This crunching into tokens means that you cannot FIND part of a keyword.

```
FIND TH
```

will not FIND all the lines containing THEN but only lines 20, 40, 60, 70 , 90 (if you have corrected it) and 100.

If you are going to restrict the range of lines to be examined when using the un-tokenised form of FIND you must remember to place both the comma and the line numbers after the closing quotes. Also remember that the line:

```
IFX>0THEN20
```

is not the same as the line:

```
IF X>0 THEN 20
```

since spaces are significant to FIND.

* Experiment with FIND, in both tokenised and un-tokenised forms until you understand why it lists some lines and not others depending upon what you type and whether you use quotes or not.

Having altered the error in line 90 we can proceed to check the rest of the program. You will probably have answered YES to the question in line 100 already, so now enter NO and press RETURN. You should find that the program prints an error message and stops. Now you can use the fifth command in the VICKIT: HELP.

HELP The HELP command is only of any use when you have just had an error

reported by the VIC-20. Type:

HELP

and press RETURN. The line in error will be listed on the screen and the section of that line in error highlighted in reverse field. The HELP command must be the first thing executed after an error otherwise the information needed by the VICKIT will be destroyed or altered.

You may find that the highlighted section of the line listed may be one character before the section of the program line which actually caused the error, but the information given by the VICKIT, together with that given by the VIC-20, should be enough to HELP you locate the error. Note that if a BASIC keyword is in error the entire keyword will be highlighted.

The sixth command that the VICKIT provides is really only useful when used in conjunction with TRACE and STEP so we will ignore OFF for the moment and return to it later. The seventh command is RENUMBER:

RENUMBER

RENUMBER is the VICKIT command which, as its name suggests, RENUMBERs the line numbers in your program. In order that the program still works it also RENUMBERs the destination line numbers in GOTOs, GOSUBs, LISTs, RUNs and IF...THENs.

There are 6 forms of the RENUMBER command ranging from a simple command to RENUMBER the entire program starting at 10 with a step line number of 10 to a version that lets you specify the range of line numbers to be RENUMBERed, the starting line number and the line number step.

The simplest form is the command:

RENUMBER

followed by RETURN, which, since no range of line numbers has been given, RENUMBERs the entire program in memory, starting at line 10 and with a step of 10 between lines. Since the original program started at 10 and had a step of 10 between line numbers there should be no change in the program, except that if you have not corrected the error in line 120 the line number to GOTO if ANS$ was "NO" would have become 0. If when RENUMBERing a program, or section of program, the VICKIT discovers a reference to a non-existent line number that reference is replaced by a 0. You can use FIND to FIND any errors caused this way by giving the command:

FIND " 0"

and pressing RETURN. Note that this will only work if you follow every GOTO, GOSUB or IF...THEN with a space.

The second form of RENUMBER allows you to specify the starting line number. This form is:

RENUMBER 100

and press RETURN. The very first line of the RENUMBERed program will be 100, the second 110, the third 120 and so on.

The third form of RENUMBER allows you to specify both the starting line number and the line number step. The step number is separated from the start line number by a comma, e.g.

RENUMBER 200,5

will RENUMBER the entire program starting at line 200 with step between line numbers of 5.

The last three forms are really variations of the same basic theme,

which allow you to specify the range of line numbers that RENUMBE will operate on. This means that RENUMBER can be made to RENUMBER sections of your program rather than the entire program. As with LIST, DELETE and FIND the range can be specified in one of three ways as follows:

<lowest line number>-< highest line number>
<lowest line number>-
-<highest line number>

the first form ranging from the first number to the second number inclusive, the second form from the first number to the end of the program and the third form from the beginning of the program to the second number inclusive. The three forms of RENUMBER are illustrated by:

RENUMBER 1000,10,1000-1999
RENUMBER 1000,10,1000-
RENUMBER 10,10,-999

By specifying the range of lines which RENUMBER is to work with you can preserve the structure of your program, leaving subroutines to start at easily remembered lines, e.g. 1000. However these three forms can be damaging, both to your sanity and your program. For example, consider the following short program:

* 10 I=1
20 PRINT I
30 I=I+1:IF I<100 THEN 20

If we apply the following command to this program:

RENUMBER 40,10,20-20

the result is:

10 I=1
40 PRINT I
30 I=I+1:IF I<100 THEN 40

The program looks correct, if a little strange, but in fact will not RUN, since the BASIC interpreter will report an ?UNDEFINED STATEMENT error in line 30.

The moral of this is that you should be extremely careful when using the restricted range form of RENUMBER since no error checking is built in to the RENUMBER routine in the VICKIT and also to save a copy of your program before you attempt this sort of RENUMBER on a program.

The final three commands best dealt with in one fell swoop since they so closely relate; STEP, TRACE and OFF. All three will be used during the debugging of a program since TRACE and STEP allow you to monitor the progress of your program while it is actually RUNning by displaying the line numbers of the last 5 lines executed by the VIC-20. Thus you can see which way your program is 'going' by finding out which lines are being executed.

TRACE and STEP work in basically the same way and are started in the same way. Decide which one of the options fits the particular circumstances of your program, type its name and press RETURN. When next you RUN the program the option selected will come into action. Once the program has been debugged the chosen option can be switched OFF by using the OFF command.

STEP Typing:

STEP

and pressing RETURN starts STEP working when the program is next RUN. When the program is RUNning the numbers of the last 5 lines executed will be displayed in the top right of the screen. Before each statement is executed by the VIC-20 the VICKIT will test the left hand shift key. If it is depressed then the statement will be

executed. If not then execution of any further statements will be held up until either the left hand shift key is depressed or the STOP key is depressed.

TRACE Typing:

TRACE

and pressing RETURN starts TRACE working in basically the same way as STEP except that the left hand shift key now acts to slow the rate at which the VIC-20 executes the lines of your program rather than speed it up as with STEP.

OFF Typing:

OFF

and pressing RETURN simply cancels either STEP or TRACE depending on which was last selected. RUNning the program then proceeds as normal.

Note that with both STEP and TRACE the line numbers are displayed in black on white in the top left hand corner of the screen. For certain colour combinations these numbers may become invisible. Also note that any INPUT statements which take effect in the top 5 lines of the screen may INPUT the line numbers there. Finally the effect of RUNning either STEP or TRACE may result in the STOP key becoming a little sluggish, so that it may have to be held down for longer than normal.

Differences between the VICKIT and the BASIC Programmer's Toolkit

APPEND There is no APPEND routine available in the VICKIT.

AUTO The default starting line number is 10 not 100.
Line numbers will only be generated by AUTO in the period between enabling AUTO and disabling it. Screen editing will not produce un-wanted line numbers and AUTO will not remember the last line number it generated or the last line number step used.

DELETE DELETE n-m, where n is greater than m, corrupts the program. DELETE n
* DELETEs from line n to the end of the program.

DUMP DUMPing will stop after each line printed until the space bar is pressed.

FIND A space must follow the keyword FIND.
Enclosing a character string within quotes does not restrict the search to strings within quotes in the BASIC program. Thus FIND "A" will FIND A= as well as PRINT "ANOTHER". The quotes are needed only to protect the character string from being crunched to its token form.
Listing of lines stops after each line until the space bar is pressed.

HELP HELP is linked in to the error vector so that pressing STOP will not set up the information needed by HELP.

OFF No known differences.

RENUMBER
Non-existent line numbers are converted to 0 not 63999.
* Setting either the start or step line numbers to 0 will result in the VICKIT corrupting the program.

STEP No known differences.

TRACE No known differences.

Note that for all the VICKIT commands the abbreviated forms available to other VIC-20 commands, e.g. V shift E for VERIFY, is not available.

Memory usage

Locations $03ED to $3FF in the first cassette buffer are used by the VICKIT together with certain page zero locations normally used by the cassette handling software in the VIC-20. This means that attempting to TRACE programs which are writing to cassette will probably cause problems.

CAUTION

* Programs that have had any of the BASIC pointers altered in page zero or any portion of themselves converted to non-standard BASIC format, e.g. by including machine code, will probably cause problems for parts of the VICKIT.

Quick reference

This section gives the permissable forms of all the VICKIT commands and can be used as a quick reference guide.

AUTO		AUTO
	or	AUTO n
	or	AUTO n,m
		where n is the starting line number and m is the step line number.
DELETE		DELETE n
	or	DELETE n-
	or	DELETE -m
	or	DELETE n-m
		where n is the lowest line number and m is the highest line number to be deleted.
DUMP		DUMP
FIND		FIND c
	or	FIND c,n-
	or	FIND c,-m
	or	FIND c,n-m
		where c is either a character string in quotes, e.g. "DATE" or not, e.g. PRINT. Note that a space must follow FIND.
HELP		HELP
OFF		OFF
RENUMBER		RENUMBER
	or	RENUMBER n
	or	RENUMBER n,m
	or	RENUMBER n,m,p-
	or	RENUMBER n,m,-q
	or	RENUMBER n,m,p-q
		where n is the starting line number for the RENUMBERed program, m is the line number step in the RENUMBERed program, p is the first line number to be RENUMBERed and q is the last line number to be RENUMBERed. Note that n must not be 0.
STEP		STEP
TRACE		TRACE

Error messages and default values

The only new error message added by the VICKIT to the VIC-20 is ?OUT OF RANGE which is given by AUTO if the line number generated would exceed 63999, and by RENUMBER if a RENUMBERed line number would be greater than 63999.

?SYNTAX will be reported by the VICKIT if any command given cannot be made to fit into the range of forms for the given command.

?OUT OF MEMORY will be reported by RENUMBER should the program being RENUMBERed expand so as to exceed the memory available.

The default value of n and m in both AUTO and RENUMBER is 10.